Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Iiro Hakkarainen

Potilastietojärjestelmän hoitomerkinnän kirjoitustyönkulun kehitys

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Hyvinvointi- ja terveysteknologia

Insinöörityö

4.5.2020

Tekijä Otsikko Sivumäärä Aika	Iiro Hakkarainen Potilastietojärjestelmän hoitomerkinnän kirjoitustyönkulun kehitys 28 sivua 4.5.2020
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikan tutkinto-ohjelma
Ammatillinen pääaine	Hyvinvointi- ja terveysteknologia
Ohjaajat	Yliopettaja Mikael Soini, Metropolia ammattikorkeakoulu Sovelluspäällikkö Mia Lundell-Viiri

Insinöörityön tavoitteena oli kehittää potilastietojärjestelmän hoitomerkinnän kirjoitustyönkulkua sellaisissa kontakteissa, joissa potilas jättää saapumatta vastaanotolle. Insinöörityö tehtiin Yritys X:lle, joka toimittaa asiakkailleen sähköistä asiakas- ja potilastietojärjestelmää. Työn kohteena oleva potilastietojärjestelmä on rakenteiseen kirjaamiseen perustuva järjestelmä, jossa on myös toiminnanohjausjärjestelmän ominaisuudet.

Työn aikana tutustuttiin potilastietojärjestelmiin, rakenteiseen kirjaamiseen ja Potilastiedon arkistoon. Lisäksi tutustuttiin malli-näkymä-ohjain-ohjelmistoarkkitehtuuriin ja tapahtumankäsittelyyn.

Työn tuloksena esiteltiin kaksi työnkulkua. Toinen työnkulku on mahdollista toteuttaa järjestelmään konfiguraatiomuutosten avulla. Tämä työnkulku konfiguroitiin testiympäristöön ja sen toiminta testattiin. Toisen esitellyn työnkulun toteutus vaatii järjestelmän toimintojen kehittämistä.

Työn tuloksia voidaan hyödyntää valittaessa käytettävää hoitotyön kirjoitustyönkulkua.

Avainsanat	Sähköinen potilastietojärjestelmä, Rakenteinen kirjaaminen

Author	Iiro Hakkarainen
Title	Improving Workflow for Progress Note Writing in EMR System
Number of Pages	28 pages
Date	4 May 2020
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Health Technology
Instructors	Mikael Soini, Principal Lecturer Mia Lundell-Viiri, Application Lead

The goal of the thesis was to improve the workflow for progress note writing in cases where the patient did not arrive to the scheduled appointment. The study was made for Company X, which provides Electronic Medical Record (EMR) for its customers. The electronic medical record is based on structured documentation. The system also has features from the Electronic Resource Planning (ERP) system.

EMR systems, structured documentation and Patient Data Repository are introduced in the paper, as well as the model-view-controller (MVC) software architecture and event handling.

The goal was to develop two solutions: One workflow which can be configured to the EMR system without software development. The other goal was to find the optimal workflow that can be implemented to the system after software development.

These workflows are presented as results. The configurable workflow was implemented into the testing environment and the workflow was tested. The other workflow would require software development. The outcome of the study can be of help when deciding the desired workflow to use.

Keywords	Electronic Medical Record, Structured documentation

Sisällys

1 Johdanto 1

2 Sähköinen potilastietojärjestelmä ja potilasasiakirjat 2

2.1 Sähköinen potilastietojärjestelmä 2
2.2 Yleiset vaatimukset tietojärjestelmille 2
2.3 Potilastiedon arkisto 3
2.4 Potilasasiakirjat 4
2.5 Palvelutapahtumat ja kontaktit 4
2.6 Rakenteinen kirjaaminen 6

3 Tapahtumankäsittely ja malli-näkymä-ohjain-arkkitehtuuri 8

3.1 Tapahtumankäsittely 8
3.2 Malli-näkymä-ohjain-arkkitehtuuri 11

4 Ongelman kuvaus ja lähtökohdat 12

4.1 Ongelman kuvaus 12
4.2 Tavoite, rajaus ja menetelmät 14

5 Vastaanottokäyntien kirjaaminen 15

5.1 Kontaktit 15
5.2 Työtila 16
5.3 Esikirjaus 17
5.4 Kirjausten kuittaaminen 17

6 Tulokset ja yhteenveto 18

6.1 Konfiguroitavat työnkulut 18
6.2 Konfiguroitavan työnkulun testaus 22
6.3 Järjestelmän kehitystä vaativa työnkulku 24
6.4 Yhteenveto 26

Lähteet 27

1 Johdanto

Insinöörityö tehtiin Yritys X:lle, joka toimittaa asiakkailleen sähköistä asiakas- ja potilastietojärjestelmää. Työn kohteena oleva potilastietojärjestelmä on rakenteiseen kirjaamiseen perustuva järjestelmä, jossa on myös toiminnanohjausjärjestelmän ominaisuudet. Rakenteisella kirjaamisella tarkoitetaan asiakas- ja potilastietojen kirjaamista ja tallentamista etukäteen sovittujen rakenteiden avulla. Tiedon rakenteistamisen apuna voidaan käyttää esimerkiksi erilaisia luokituksia, koodistoja tai yhtenäistä termistöä. Rakenteisesti kirjattu tieto mahdollistaa tehokkaamman tiedon vaihtamisen eri järjestelmien tai organisaatioiden välillä ja tehokkaamman tiedon koneellisen käsittelyn.

Insinöörityön aiheena on kehittää potilastietojärjestelmän hoitomerkinnän kirjoitustyönkulkua aikataulutetuille käynneille saapumatta jättäneille asiakkaille. Nykyisessä työnkulussa, kun asiakas jättää saapumatta vastaanotolle, aikataulutettuun kontaktiin ei kirjata hoitoon liittyviä jatkosuunnitelmia tai muuta tietoa, vaan luodaan uusi aikatauluttamaton kontakti, johon merkintä kirjataan. Työnkulusta tulleen palautteen perusteella on todettu, että työnkulku ei ole intuitiivinen. Insinöörityössä on tarkoituksena löytää ratkaisu tähän ongelmaan, eli kehittää intuitiivisempi työnkulku.

Työssä tutustutaan sähköisiin potilastietojärjestelmiin, rakenteiseen kirjaamiseen ja Potilastiedon arkistoon. Työssä tutustutaan lisäksi malli-näkymä-ohjain-ohjelmistoarkkitehtuuriin sekä tapahtumankäsittelyyn.

Työn tavoitteena on löytää kaksi ratkaisua: ratkaisu, joka voidaan toteuttaa nykyisellään konfiguraatiomuutosten avulla, sekä ratkaisu, joka olisi mahdollista toteuttaa järjestelmän kehitystyön jälkeen.

2 Sähköinen potilastietojärjestelmä ja potilasasiakirjat

2.1 Sähköinen potilastietojärjestelmä

Sähköinen potilastietojärjestelmä on ohjelmisto tai järjestelmä, jolla tallennetaan ja käsitellään asiakas- ja potilastietoja [1]. Terveydenhuollon palveluntuottajat kirjaavat tietojärjestelmään potilastietoja ja muita hoidon järjestämisen ja toteuttamisen yhteydessä laadittuja tai saatuja tietoja tai asiakirjoja. Näistä usein potilaan terveydentilaa koskevia tietoja sisältävistä tiedosta käytetään nimitystä *potilasasiakirja*. Potilasasiakirjat muodostavat *potilaskertomuksen*, eli aikajärjestyksessä etenevän potilasasiakirjan [2].

Sosiaali- ja terveydenhuollon tietojärjestelmät jakautuvat kahteen luokkaan. Luokkaan A kuuluvat Kansaneläkelaitoksen Kanta-palvelut ja Kanta-palveluihin suoraan tai välityspalvelun kautta liitettävät potilastietojärjestelmät. Luokkaan B kuuluvat kaikki muut, alueellisesti tai paikallisesti käytettävät tietojärjestelmät. [1.]

2.2 Yleiset vaatimukset tietojärjestelmille

Yleiset vaatimukset tietojärjestelmille, niiden valmistajille ja sosiaali- ja terveydenhuollon palvelun antajille on määritelty sosiaali- ja terveydenhuollon asiakastietojen sähköisestä käsittelystä annetussa laissa (159/2007). Lisäksi Terveyden ja hyvinvoinnin laitoksen (THL) määräyksissä annetaan tietojärjestelmille yksityiskohtaiset olennaiset vaatimukset. Valvira ylläpitää rekisteriä vaatimukset täyttävistä tietojärjestelmistä sekä vastaanottaa ja käsittelee potilasturvallisuutta, tietoturvaa ja tietosuojaa merkittävästi vaarantavista poikkeamista. Valviralla on myös oikeus tehdä valvonnan edellyttämiä tarkastuksia. [1.]

2.3 Potilastiedon arkisto

Potilastiedon arkisto on Sosiaali- ja terveydenhuollon asiakastietojen sähköisestä käsittelystä annetun lain (159/2007) 14 §:n 1 momentissa tarkoitettu valtakunnallinen arkistointipalvelu.

Potilastiedon arkisto kuuluu osaksi valtakunnallisia terveydenhuollon tietojärjestelmäpalveluita eli Kanta-palveluita. Potilastiedon arkisto käsittää Tiedonhallintapalvelun ja potilasasiakirjojen arkistointipalvelun. Tiedonhallintapalveluun kuuluvat potilaan kieltojen, suostumusten, tahdonilmaisujen sekä keskeisten terveystietojen hallintapalvelut. Omakannan kautta kansalaiset voivat katsoa Potilastiedon arkistoon tallennettuja potilastietoja ja sähköisiä lääkemääräyksiä. Asiakirjojen arkistointia ei voi kieltää, eikä arkistointi vaadi potilaan suostumusta tai informointia. [3.]

Potilastiedon arkistoon tallennetaan potilastiedot yhtenevässä muodossa, mikä parantaa tietojen siirrettävyyttä ja saatavuutta eri terveydenhuollon palveluntarjoajien välillä. Arkisto on keskeisessä roolissa asiakas- ja potilastietojen välittämisessä palveluntarjoajien kesken. [4.]

Sosiaali- ja terveydenhuollon asiakastietojen sähköisestä käsittelystä annetussa laissa annetaan THL:lle tehtäväksi määrittää valtakunnallisten tietojärjestelmäpalvelujen toteutuksen edellyttämät tietosisällöt, käsitemallit ja toimintaprosesseja tukevat tietorakenteet. Kansaneläkelaitoksen toteuttama ja ylläpitämä koodistopalvelu sisältää kaikki koodistot, joita tarvitaan asiakasasiakirjojen käsittelyssä valtakunnallisten tietojärjestelmäpalvelujen avulla. THL vastaa koodistopalvelun sisällöstä. [5.]

2.4 Potilasasiakirjat

Sosiaali- ja terveysministeriön asetus potilasasiakirjoista määritellee potilasasiakirjoihin kuuluvaksi potilaskertomuksen ja siihen liittyvät potilastiedot tai asiakirjat sekä lääketieteelliseen kuolemansyyn selvittämiseen liittyvät tiedot tai asiakirjat samoin kuin muut potilaan hoidon järjestämisen ja toteuttamisen yhteydessä syntyneet tai muualta saadut tiedot ja asiakirjat. Asetuksen mukaan tallennettava potilasasiakirja voi sisältää potilaan hoitoon tai hoitoon liittyviin tehtäviin osallistuvien henkilöiden tekemiä merkintöjä.

Potilasasiakirjoja koskevan asetuksen mukaan potilasasiakirjat tallennetaan sähköisinä Potilastiedon arkistoon, sellaisina teknisinä tallenteina, jotka koostuvat yhteen palvelutapahtumaan liitetyistä merkinnöistä, jotka jakavat saman säilytysajan ja tietoteknisen tallennusmuodon. Potilastiedon arkistoon tallennettavien sähköisten potilasasiakirjojen tulee muodostaa ehyt asiakirjakokonaisuus yksilöityjen palvelutapahtuma- ja palvelukokonaisuustunnusten avulla. [2.]

Palvelutapahtumalla tarkoitetaan Sosiaali- ja terveydenhuollon asiakastietojen sähköisestä käsittelystä annetun lain (159/2007) 3 §:n 5 kohdan määrittelyn mukaan terveydenhuollon palvelujen antajan ja potilaan välistä yksittäisen palvelun järjestämistä tai toteuttamista. [5.]

Potilasasiakirjoihin on merkittävä selkeästi ja ymmärrettävästi tarpeelliset ja riittävät tiedot hyvän hoidon suunnittelun, järjestämisen, toteuttamisen ja seurannan turvaamiseksi. Potilasasiakirjat on ennen kaikkea tarkoitettu tukemaan potilaan hyvää hoitoa. [6.]

2.5 Palvelutapahtumat ja kontaktit

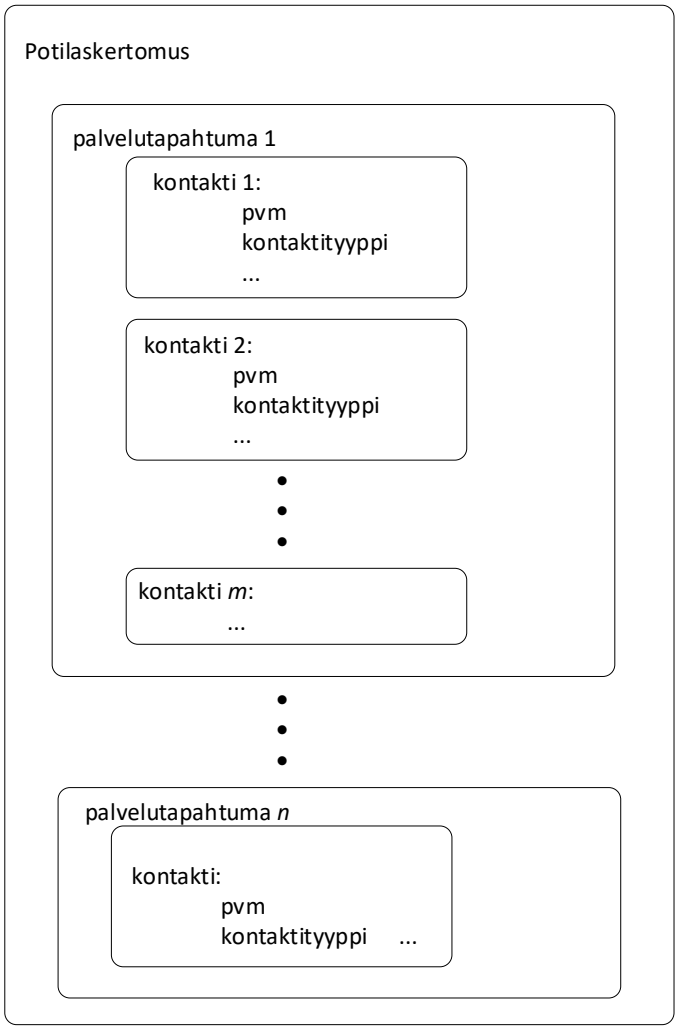
Luvussa 2.4 esiteltiin palvelutapahtuma, eli yksittäinen terveydenhuollon palvelujen antajan ja potilaan välinen palvelun järjestäminen tai toteutuminen. Palvelutapahtuma-asiakirja ja palvelutapahtuman tunniste luodaan teknisesti potilastietojärjestelmään potilaan ottaessa yhteyttä, esimerkiksi varaamalla ajan tai saapumalla päivystysvastaanotolle [3]. Palvelutapahtuma tarkastelee hoitoa potilaan eikä palveluntuottajan näkökulmasta.

Vaikka potilaan hoidon aikana koordinaatiovastuu siirtyisi osastolta toiselle, näistä muodostuu silti vain yksi palvelutapahtuma, mikäli hoidon syy pysyy ennallaan [7].

Insinöörityössä käsiteltävässä potilastietojärjestelmässä yksittäisestä hoitokerrasta muodostuu *kontakti*, johon tallennetaan kyseisen hoitotapahtuman kirjaukset. Jokainen kontakti on liitetty palvelutapahtumaan. Kontakti voi olla Potilastiedon arkiston mukainen palvelutapahtuma tai yksittäinen hoitoprosessin tapahtuma osana palvelutapahtumaa, eli prosessitapahtuma. Kontakteilla on siis mahdollista jaotella palvelutapahtuma potilastietojärjestelmässä pienempiin kokonaisuuksiin.

Yksi kontakti voi olla esimerkiksi avovastaanoton käynti tai vuodeosastopotilaan käynti toimenpideyksikössä. Ensimmäinen esimerkeistä on kontakti, joka itsessään muodostaa palvelutapahtuman. Jälkimmäinen esimerkki on palvelutapahtuman prosessitapahtuma, sama palvelutapahtuma sisältäisi osastohoitojakson sekä toimenpidekäynnin.

Potilaan hoitoasiakirjat sisältävä potilaskertomus koostuu siis *n* määrästä palvelutapahtumia, jotka puolestaan koostuvat *m* määrästä kontakteja. Kuvassa 1 on kuvattu edellä esitetty potilaskertomuksen tietorakenne. [7.]



Kuva 1. Potilaan hoitokertomuksen jakaminen palvelutapahtumiin ja kontakteihin.

Kuvassa 1 on esitetty potilaskertomus, potilaskertomuksen jakautuminen palvelutapahtumiin ja palvelutapahtumien jakautumisen edelleen kontakteihin.

2.6 Rakenteinen kirjaaminen

Rakenteisella kirjaamisella tarkoitetaan asiakas- ja potilastietojen kirjaamista ja tallentamista etukäteen sovittujen rakenteiden avulla. Rakenteisesti kirjattu tieto mahdollistaa tehokkaamman tiedon koneellisen käsittelyn ja tiedon vaihtamisen eri järjestelmien tai organisaatioiden välillä. Tiedon rakenteistamisen apuna voidaan käyttää esimerkiksi erilaisia luokituksia, koodistoja tai yhtenäistä termistöä [8].

Potilaskertomuksen rakenteistamisella on tarkoitus yhtenäistää kirjaamista sekä helpottaa potilastietojen käyttöä, hakua ja hoidon seurantaa. Vapaamuotoisesti kirjoitetun tekstin sisältämän tiedon hyödynnettävyys on rajallisempi kuin rakenteisen tiedon. Suurin hyöty rakenteisesta kirjaamisesta tulee siitä, että tieto tarvitsee kirjata vain kerran, jonka jälkeen se on hyödynnettävissä tietojärjestelmässä tai toisessa organisaatiossa.

Tietojen kirjaaminen yhtenevällä tavalla parantaa potilastiedon laatua ja edistää potilasturvallisuutta. Kirjaaminen sovittujen tietorakenteiden mukaan mahdollistaa myös palvelujen toteutumisen arvioinnin ja toiminnan ohjauksen aikaisempaa paremmin raportoinnin helpomman automatisoinnin myötä. Myös lakiperustaiset valvonta- ja seurantatehtävät helpottuvat.

Taulukossa 1 on kuvattu esimerkki rakenteisesta kirjaamisesta. Rakenteisesti kirjattu tieto on kirjoitettu vapaamuotoisena tekstinä taulukon alle. Diagnoosin kirjaamiseen käytetään kansainvälisen tautiluokitusjärjestelmän, ICD-10 (International Statistical Classification of Diseases and Related Health Problems) luokituksen koodeja.

Taulukko 1. Yksinkertaistettu esimerkki rakenteisesta kirjaamisesta.

Tieto	Arvo
Nimi	Raimo Rakenteinen
Ikä	50 vuotta
Oireet	Väsymys, laihtuminen, lisääntynyt virtsaneritys
Diagnoosi (ICD-10)	E14.9
Diagnoosin varmuusaste	Epäilty
Mitattu verensokeri	9 mmol/l
Lähetteet	1483 Pt-Gluk-R1

Taulukon rakenteinen tieto voisi olla kirjattuna vapaaksi tekstiksi seuraavalla tavalla:

> Raimo Rakenteinen, 50-vuotias mies. Viimeaikaisia oireita väsymys, laihtuminen ja virtsanerityksen lisääntyminen. Epäily aikuistyypin diabeteksesta (E14.9). Vastaanotolla mitattu verensokeri 9 mmol/l. Kirjoitettu lähete sokerirasituskokeeseen.

Edistyneiden tekoälyä hyödyntävien ohjelmien avulla esimerkin vapaata tekstiä voitaisiin mahdollisesti käsitellä myös koneellisesti, mutta rakenteisen tiedon käsittely on tehokkaampaa, eikä jätä tulkinnanvaraa.

Rakenteinen kirjaaminen varmistaa yhdenmukaisen kirjaamisen ja laadullisesti hyvän ja kattavan asiakas- ja potilastiedon. Aikaisemmin kirjatun tiedon hyödyntäminen helpottuu, koska tietoa on helpompi hakea. Tämä tuo ammattihenkilöille hyötyä päivittäiseen työskentelyyn. [8.]

3 Tapahtumankäsittely ja malli-näkymä-ohjain-arkkitehtuuri

3.1 Tapahtumankäsittely

Tapahtumaohjatulla arkkitehtuurilla (Event-driven architecture) tarkoitetaan mallia, jossa ohjelmalla on kyky havaita tapahtumia ja suorittaa toimintoja tapahtumien perusteella. Tietokoneohjelmistoissa *tapahtuma* on usein tilan muutos, esimerkiksi jonkin muuttujan arvon muutos. Tapahtuma voi olla myös esimerkiksi näppäimen painallus. Reaalimaailman esimerkki tapahtumasta on esimerkiksi nukahtaminen tai ikkunan ohi lentävä pallo. Tapahtuma voi myös olla odotettava toiminto, joka ei toteudukaan. [9.]

Tapahtuma voidaan siis mieltää ohjelmaa suoritettaessa syntyväksi tilaksi, jonka perusteella ohjelmassa suoritetaan tai ollaan suorittamatta jokin toiminto. Tapahtuman seurauksena voidaan esimerkiksi suorittaa aliohjelma tai näyttää tietoa näytöllä. Ohjelmistoissa tapahtumia kuvataan usein viesti-nimisellä tietorakenteella. Viesti sisältää usein tiedon tapahtuman tyypistä sekä mahdollisia parametrejä tapahtuman käsittelyä varten. [10.]

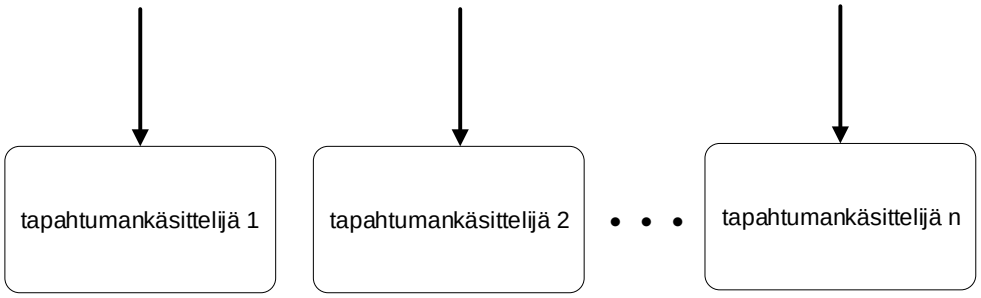
Tapahtumia käsitteleviä ohjelman osia kutsutaan tapahtumankäsittelijöiksi. Tapahtumankäsittelijät toimivat samankaltaisesti kuin esimerkkikoodissa 1 esitetyt olio-ohjelmoinnissa käytettävien luokkien metodit.

```
minunOlio = new Luokka()

minunOlio.set(“abc”)
print minunOlio.get() # tulostaa “abc”
```

Esimerkkikoodi 1. Pseudokoodilla kuvattu olion luonti ja metodien käyttö.

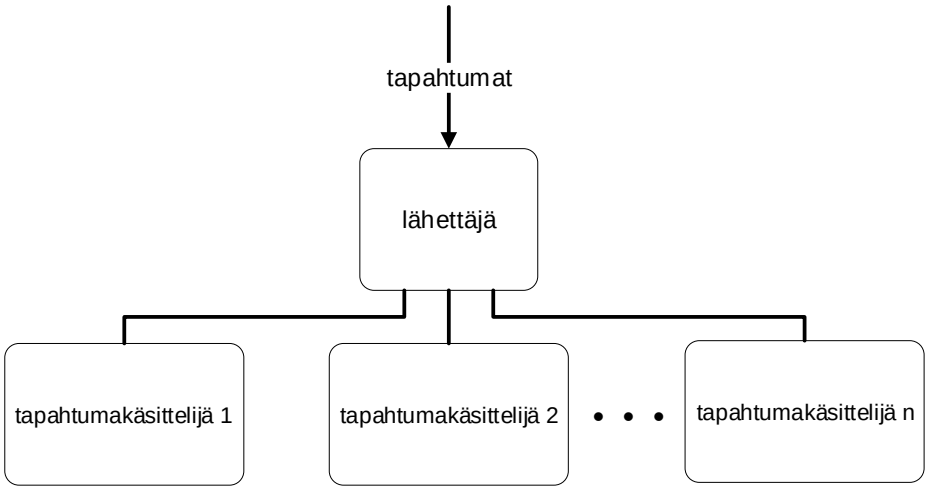
Esimerkkikoodissa 1 luodaan luokan *Luokka*-olio. *Set*-metodia käytetään asettamaan oliolle arvo ja olion arvo tulostetaan hakemalla arvo *get* -metodilla. Luokka on ikään kuin kokoelma tapahtumankäsittelijöitä, joilla tässä tapauksessa käsitellään *set*- ja *get*-tapahtumia. [11.]

Esimerkkikoodissa tapahtumat käsitellään ilman lähettäjää. Kuvassa 2 on esitetty ilman lähettäjää tehtävä tapahtumankäsittely.



Kuva 2. Tapahtumankäsittely ilman lähettäjää. (mukaillen [11])

Koodin ylläpidettävyyden ja uudelleenkäytettävyyden sekä useamman tapahtuman samanaikaisen tai peräkkäisen käsittelyn helpottamiseksi tapahtumankäsittelyyn voidaan lisätä lähettäjä. Lähettäjä vastaanottaa tapahtumasta viestin, joka sisältää tapahtumatyypin sekä mahdolliset parametrit tapahtuman käsittelyä varten. Kuvassa 3 on kuvattu tapahtumankäsittelijä, jossa on lähettäjä.



Kuva 3. Tapahtumankäsittely lähettäjän kanssa. (mukaillen [11])

Kuvassa 3 tapahtumat saapuvat lähettäjälle, joka lähettää tapahtumat eteenpäin kyseistä tapahtumatyyppiä kuuntelevalle tapahtumankäsittelijälle. Lähettäjä voi joutua käsittelemään monta tapahtumaa peräkkäin, joten toistorakenteen käyttö on suotavaa. Toistorakennetta, esimerkiksi *While*-silmukkaa hyödyntämällä, lähettäjä voi vastaanottaa tapahtuman, lähettää sen tapahtumankäsittelijälle ja siirtyä seuraavan tapahtuman prosessointiin. Tämä on kuvattu esimerkkikoodissa 2.

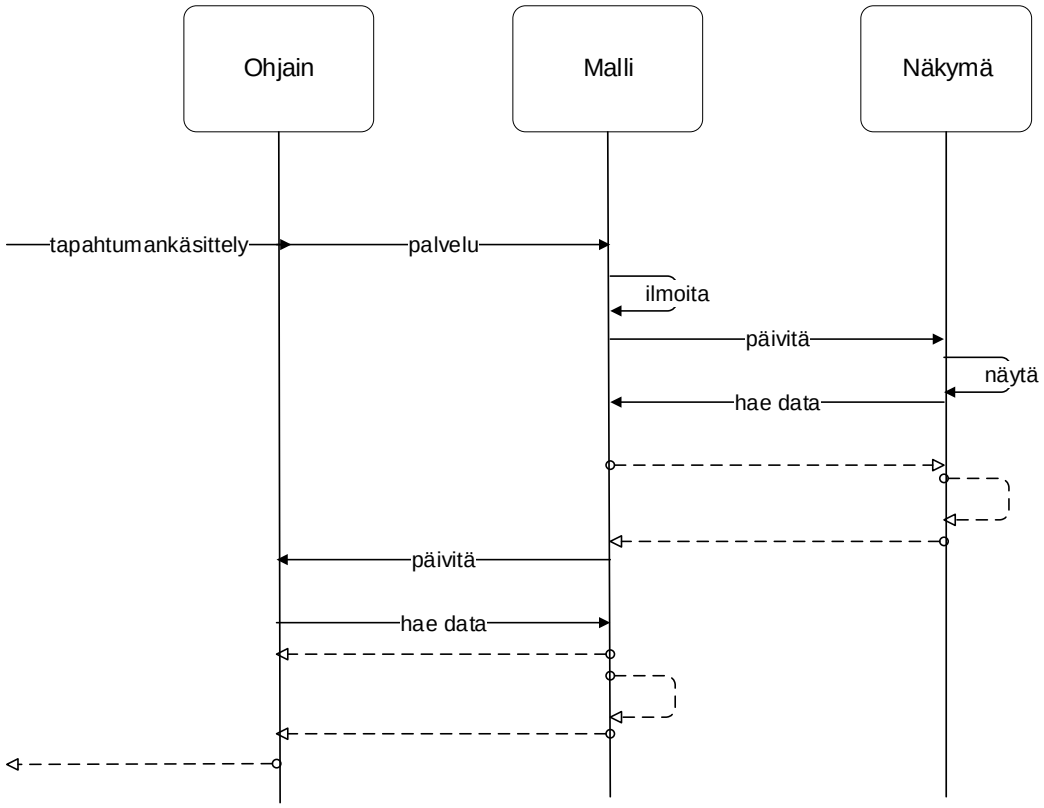
```
While 1:  #tapahtumasilmukka
 tapahtuma = haeSeuraavaTapahtumaVirrasta()

 if tapahtuma.tapahtumaTyyppi == TapahtumaVirranLoppu:
      break

 if tapahtuma.tapahtumaTyyppi == 1:
      kutsutaan haluttua tapahtumankäsittelijää

 if tapahtuma.tapahtumaTyyppi == 2:
      kutsutaan haluttua tapahtumankäsittelijää

 else: #tuntemattoman tapahtumatyypin käsittely
```

Esimerkkikoodi 2. Tapahtumasilmukka tapahtumavirran käsittelyyn.

Esimerkkikoodissa 2 on pseudokoodilla kuvattu tapahtumasilmukka, jossa tapahtumat lähetetään tyypin mukaan tapahtumankäsittelijälle. Tapahtumatyypin ollessa *TapahtumaVirranLoppu* silmukan suorittaminen keskeytyy. Else-ehdossa käsitellään tuntemattomat tapahtumatyypit.

3.2 Malli-näkymä-ohjain-arkkitehtuuri

Malli-näkymä-ohjain-arkkitehtuuri (Model-view-controller, MVC) koostuu kolmesta komponentista, joiden avulla käyttöliittymä erotetaan sovelluslogiikasta ja datasta. MVC-arkkitehtuurin avulla sovelluksen ylläpidettävyys ja järjestelmän siirrettävyys toiselle käyttöliittymälle helpottuu. Jokaisella komponentilla on oma tehtävänsä: Ohjaimet (Controller) vastaanottavat syötteitä käyttäjältä ja järjestelmästä ja nimensä mukaan ohjaa (tai käskee) mallia ja näkymää muuttumaan tarvittaessa. Mallit (Model) kuvaavat sovelluksen tilaa eli edustavat jotakin osaa sovellusdatasta. Näkymät (View) edustavat jotain osaa näkyvästä käyttöliittymästä. Riippuen sovelluksesta samaan malliin saattaa liittyä useampia näkymiä. Kuvassa 4 on kuvattu MVC-arkkitehtuurin toiminta.



Kuva 4. Malli-näkymä-ohjain-arkkitehtuurin toiminta. (mukaillen [12])

Kuvan 4 mallissa vuorovaikutus alkaa, kun käyttäjä tekee jonkin komennon järjestelmässä. Ohjain havaitsee tämän ja pyytää mallia suorittamaan komentoa vastaavan loo-

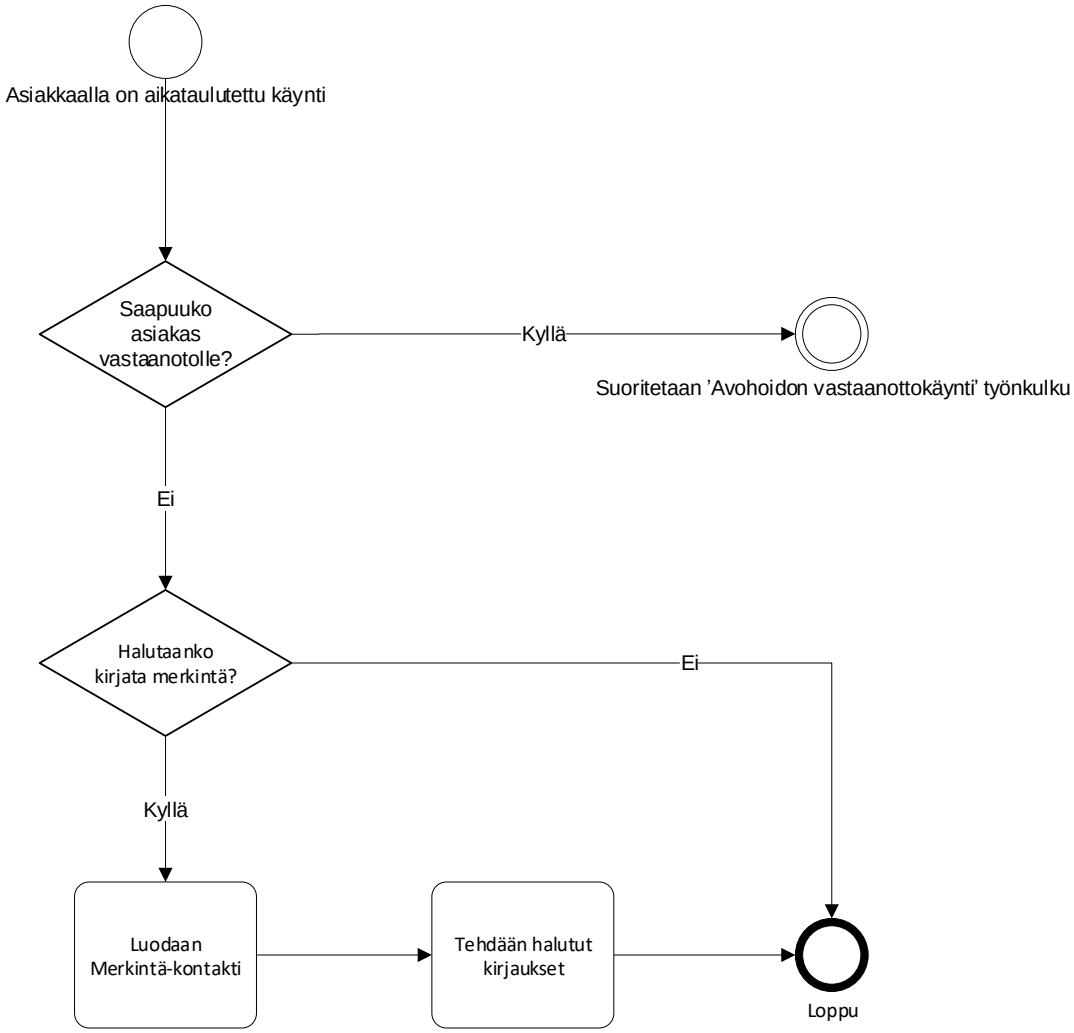
gisen sovelluspalvelun. Mallin tila muuttuu komennon seurauksena, ja se informoi muutoksesta mallia kuuntelevia näkymiä ja ohjaimia. Näkymä pyytää mallilta muuttunutta tietoa ja päivittää näytön. Myös ohjain kysyy mallilta muuttuneita tietoja ja reagoi sen perusteella. Mallin tilan päivityksen myötä jokin komento saattaa esimerkiksi olla kielletty tai uusi komento on voinut tulla sallituksi ja ohjaimen täytyy tietää tämä.

MVC-arkkitehtuuri on luonnollinen valinta graafisen käyttöliittymän toteutusmekanismiksi, sillä se mahdollistaa mallin uudelleenkäytön lähes kaikissa tilanteissa. Lähes kaikki graafisen käyttöliittymän ympäristöt käyttävät jollain tavalla MVC-mallin mukaista arkkitehtuuria. Malliin liittyy myös joitain ongelmia. Tyypillisesti MVC-malli monimutkaistaa järjestelmää lisäten järjestelmän luokkia. Ohjain- ja näkymäluokat kannattaa antaa vain isoimmille käyttöliittymäkokonaisuuksille, jotta vältetään järjestelmän liiallista pirstoutumista. [12, s. 142–145.]

4 Ongelman kuvaus ja lähtökohdat

4.1 Ongelman kuvaus

Insinöörityön tavoitteena on parantaa potilastietojärjestelmän hoitomerkinnän kirjoitustyönkulkua aikataulutetuille käynneille saapumatta jättäneille asiakkaille. Työn kohteena on rakenteiseen kirjaamiseen pohjautuva potilastietojärjestelmä. Nykyisessä työnkulussa potilaan jättäessä saapumatta aikataulutetulle käynnille kontaktiin ei kirjata hoitoon liittyviä jatkosuunnitelmia tai muuta tietoa. Sen sijaan luodaan uusi aikatauluttamaton kontakti, johon halutut tiedot kirjataan. Tämä työnkulku on kuvattu kuvassa 5.



Kuva 5. Nykyinen työnkulku hoitomerkinnän kirjoittamiseen saapumatta jättäneelle asiakkaalle.

Työnkulku alkaa siitä, kun potilaalla on aikataulutettu vastaanottokäynti. Potilas ei saavu vastaanotolle, mutta ammattihenkilö haluaa silti kirjoittaa potilaalle merkinnän. Ammattihenkilö luo potilaalle uuden kontaktin, kirjoittaa merkinnän ja kuittaa kirjaukset.

Järjestelmän käyttäjiltä tulleen palautteen perusteella on todettu, että työnkulku ei ole intuitiivinen. Uuden kontaktin luominen on aiheuttanut käyttäjissä hämmennystä. Insinöörityössä pyritään löytämään ratkaisu tähän ongelmaan.

4.2 Tavoite, rajaus ja menetelmät

Työn tavoitetta voidaan kuvata seuraavalla tutkimuskysymyksellä: Kuinka hoitomerkinnän kirjoitustyönkulkua voidaan kehittää vastaanotolle saapumatta jättäneiden potilaiden kohdalla?

Työn tavoitteena on kehittää hoitomerkinnän kirjoittamisen työnkulkua intuitiivisemmaksi niin sanotuille ’no-show-käynneille’, eli käynneille, joissa asiakas jättää saapumatta aikataulutetulle vastaanottokäynnille.

Työssä pyritään löytämään kaksi ratkaisua:

- Ratkaisu, joka on mahdollista toteuttaa konfiguraatiomuutoksilla.
- Ratkaisu, joka on mahdollista toteuttaa kehitystyön jälkeen.

Konfiguraatiomuutoksilla tarkoitetaan muutoksia, joissa käytetään järjestelmässä jo olemassa olevia toiminnallisuuksia. Kehitystyön jälkeisillä muutoksilla tarkoitetaan muutoksia, jotka vaativat uusien toimintojen ohjelmoimista järjestelmään.

Työnkulun kehitys rajataan koskemaan *avohoidon vastaanottokäynti* -kontakteja. Lisäksi rajataan työ käsittämään vain työnkulun kehitys ja testaus. Työhön ei kuulu esimerkiksi työnkulkujen koulutus tai käyttöönotto. Rajaamisen perusteena on omien työtehtävien rajoittuminen kyseisiin tehtäviin.

Työssä kartoitetaan konfiguraatiomuutoksilla mahdollisia työnkulkuja. Työnkuluista valitaan yksi, joka konfiguroidaan järjestelmään ja jonka toimivuus testataan. Lisäksi arvioidaan, millainen sovelluskehityksen jälkeen mahdollinen työnkulku olisi optimaalinen.

5 Vastaanottokäyntien kirjaaminen

5.1 Kontaktit

Luvussa 2.5 esitettiin käsite kontakti. Järjestelmäteknisesti kontaktilla tarkoitetaan tietokantatietuetta. Potilaalla/asiakkaalla ei siis ole vain yhtä tietokantatietuetta, jota päivitetään ja lisätään, vaan jokaisesta kontaktista luodaan oma tietueensa. Näin toimimalla jokaisen yksittäisen kontaktin tietoja on helppo tarkastella ja potilaskertomuksesta muodostuu looginen kokonaisuus. Tietokantamalleissa tietueen tunniste, eli id, on pääavain ja sen on oltava uniikki. Tietueen tunnistetta ei siis voida käyttää, jos haetaan yksittäisen potilaan kaikkien tietueiden tietoja (olettaen, että kaikkien tietueiden tunnisteita ei tunneta). Taulukon 2 esimerkki kuvaa tietokantataulua, joka ei toimisi, sillä tietueiden tunnisteet eivät ole uniikkeja.

Taulukko 2. Tietokantataulu ilman uniikkia tunnistetta.

id	nimi	pvm
123	Tieto Kantanen	2018-06-12
123	Tieto Kantanen	2019-02-22

Ratkaisuna tähän voidaan käyttää esimerkiksi uutta saraketta, johon tallennetaan yksilöllinen, potilaan ensimmäisestä kontaktista periytyvä tunniste. Edellä mainittu ratkaisu on esitelty taulukossa 3. [13.]

Taulukko 3. Tietokantataulu, jossa uniikki tunniste.

id	potilas_id	nimi	pvm
123	123	Tieto Kantanen	2018-06-12
186	123	Tieto Kantanen	2019-02-22
269	123	Tieto Kantanen	2019-05-13
455	123	Tieto Kantanen	2020-01-09

Taulukon 3 mallin mukaan jokaisella potilaalla on järjestelmässä yksilöllinen tunniste ja jokaisella tietueella on uniikki pääavain.

Teknisesti kontaktit ovat siis tietokantatietueita, joihin tallennetaan yksittäisen hoitokontaktin tiedot. Kontakti liitetään palvelutapahtumaan ja sen pohjalta muodostetaan Potilastiedon arkistoon lähetettävät hoitoasiakirjat.

Tämän työn kannalta mielenkiintoinen kontakteista löytyvä kenttä on ’ajanvarauksen tila’ (jäljempänä myös ’käynnin tila’). Ajanvarauksen tilalla on eri arvoja riippuen siitä, minkälaisia kirjauksia kyseiseen kontaktiin järjestelmässä on tehty. Kun vastaanottokäynti on varattu, kontaktin tila on ’aikataulutettu’. Kun potilas kirjataan vastaanotolle, käynnin tilaksi muuttuu ’saapunut’. Kun käynti on valmis ja ammattihenkilö kuittaa käynnin kirjaukset, käynnin tilaksi muuttuu ’valmis’. Jos potilas jättää saapumatta, käynnin tilaksi voidaan kirjata ’Ei saapunut’. Käynnit, joita ei kirjata vastaanotolle muuttuu automaattisesti ’Ei saapunut’-tilaan kolmen päivän kuluttua vastaanotosta.

Toinen kiinnostava kenttä on kontaktityyppi. Kontaktityypillä kuvataan nimensä mukaan kontaktin tyyppiä, esimerkiksi puhelu tai vastaanottokäynti ovat omat tyyppinsä. Tässä työssä kiinnostavia kontaktityyppejä ovat ’ajanvaraus’ sekä ’vastaanottokäynti’. Kun käynti on varattuna ja käynnin tila on ’aikataulutettu’, kontaktityyppi on ’ajanvaraus’. Kun potilas kirjataan vastaanotolle, kontaktityypiksi muuttuu ’vastaanottokäynti’. Kaikki aikataulutetut käynnit ovat alun perin kontaktityypiltään tyyppiä ’ajanvaraus’. Kun kontaktiin tehdään tiettyjä kirjauksia, kuten vastaanotolle kirjaaminen, kontaktityyppi muuttuu erilaisten sääntöjen perusteella vastaamaan kontaktin tyyppiä – useimmiten vastaanottokäynniksi.

5.2 Työtila

Työtilalla tarkoitetaan näkymää, joka sisältää kontaktin tietojen kirjaamiseen käytettävät valikot ja toiminnot. Riippuen käyttäjän roolista, työtilat sisältävät erilaisia valikkoja ja toimintoja. Esimerkiksi sihteereillä voi olla tarvetta eri toiminnoille kuin esimerkiksi erikoislääkärillä. Kun järjestelmässä avataan kontakti, johon on tarkoitus kirjata, avautuva nä-

kymä eli työtila määräytyy erilaisten sääntöjen avulla. Sääntöjen ehtolauseet voivat esimerkiksi evaluoida käyttäjän roolia tai erikoisalaa ja eri totuusarvojen perusteella työtilaan asetetaan erilaisia toimintoja.

5.3 Esikirjaus

Esikirjaus on tila, jonka ammattihenkilö voi avata ennen kuin potilas on kirjattu vastaanotolle. Esikirjaustila sisältää rajoitetun määrän työkaluja verrattuna normaaliin vastaanotoilla käytettävään työtilaan. Esikirjaustilassa ei esimerkiksi voida kirjata peruselintoimintoja, koska järjestelmä olettaa, että potilas ei ole vielä saapunut. Kun potilas saapuu paikalle, hänet kirjataan sisään vastaanotolle järjestelmässä, jolloin saadaan kaikki tarvittavat työkalut käyttöön. Jos potilas jättää saapumatta, eli potilasta ei koskaan kirjata vastaanotolle, esikirjauksessa kirjatut tiedot poistetaan, kun ajanvarauksesta on kulunut tietty aika. Kirjaukset poistetaan, koska esikirjauksessa on voitu esimerkiksi määrätä joitain lääkkeitä tai laboratoriokokeita pitoon odottamaan potilaan saapumista vastaanotolle. Kun potilas seuraavan kerran saapuu vastaanotolle, pidossa olevat määräykset eivät välttämättä enää ole tilanteeseen sopivia.

5.4 Kirjausten kuittaaminen

Kun ammattihenkilö on kirjannut kontaktiin haluamansa tiedot, ammattihenkilön tulee kuitata käynti. Kuittaamisella ammattihenkilö varmentaa kirjausten oikeellisuuden, ikään kuin allekirjoittaa hoitoasiakirjat. Käynnin kuittaaminen myös laukaisee Kanta-sanomien generoimisen ja lähettämisen sekä käynnin laskutuksen.

Ajanvaraus-tyyppisiä kontakteja ei voida kuitata, koska niitä käsitellään teknisesti ikään kuin käynteinä, jotka eivät ole vielä toteutuneet – eli ajanvarauksina. Toisaalta, jos kontaktityyppi on ’ajanvaraus’, kontaktiin ei ole tehty kirjauksia, sillä muuten kontaktityyppi olisi muuttunut (kuten kuvattu luvun 5.1 lopussa).

6 Tulokset ja yhteenveto

6.1 Konfiguroitavat työnkulut

Uudella konfiguraatiolla toteutettavissa olevien työnkulkujen kartoituksessa tuli miettiä, mihin kaikkeen järjestelmä nykyisellään taipuu. Käytössä olevassa järjestelmän oletustyönkulussa (esitetty kuvassa 5) luodaan uusi kontakti merkinnän kirjoitusta varten. Uudessa työnkulussa on tarkoituksena saada kirjattua merkintä samaan ’Ei saapunut’ -tilassa olevaan kontaktiin. Merkinnän liittäminen samaan kontaktiin on loogisempaa, tästä kertoo esimerkiksi se, että useampi loppukäyttäjä antoi tämänsuuntaista palautetta, myös yrityksen kliiniset asiantuntijat olivat asiasta yhtä mieltä.

Järjestelmän konfiguraatiomahdollisuudet rajoittavat sitä, millaisia työnkulkuja on mahdollista rakentaa. Käytännössä kaikkia mahdollisia konfiguraatiomuutoksilla toteutettavia työnkulkuja, joissa merkintä kirjataan saapumatta jättäneen potilaan yhdistää seuraavat askeleet:

- Kontakti pitää merkitä ’Ei saapunut’ -tilaan.
- Kontaktiin tulee tehdä tarvittavat kirjaukset, jotta kontaktityyppi muuntuu ajanvarauksesta vastaanottokäynniksi.
- Kontakti tulee kuitata, jotta hoitoasiakirjat lähetetään potilastiedon arkistoon.

Kontakti kirjataan ’Ei saapunut’ -tilaan, jotta käynti saadaan tilastoitua oikein. Kontaktin tyyppi pitää muuntua ajanvarauksesta joksikin muuksi tyypiksi, jotta käynti saadaan kuitattua. Kontakti pitää kuitata, jotta laukaistaan potilastiedon arkistoon menevien hoitoasiakirjojen muodostuminen ja lähetys. Käytännössä edellä mainitut esivaatimukset työnkululle tarkoittavat sitä, että erottava tekijä mahdollisilla työnkuluilla on kontaktityypin muuntamisen laukaiseva tekijä.

Kontaktityypin muuntamisen laukaisevien tietojen kirjaamisia ovat esimerkiksi rokotteen kirjaaminen, peruselintoimintojen kirjaaminen, diagnoosin tai käyntisyyn kirjaaminen ja

tulosyyn kirjaaminen. Uuden työnkulun kannalta tuli valita sellainen kontaktityypin muuntava tekijä, joka on realistista kirjata jokaiseen ei-saapuneeksi merkattuun kontaktiin. Esimerkiksi rokotetta tai peruselintoimintoja ei varmastikaan kovinkaan usein kirjata potilaille, jotka jättävät saapumatta vastaanotolle.

Varteenotettaviksi vaihtoehdoiksi jäivät diagnoosin/käyntisyyn kirjaaminen sekä tulosyyn kirjaaminen. Tulosyyllä tarkoitetaan potilaan itse kertomaa hoitoon hakeutumisen syytä. Diagnoosilla tarkoitetaan potilaan oireiden perusteella todettua sairautta. Käyntisyyllä tarkoitetaan muun kuin lääkärin tai hammaslääkärin kirjaamaa tietoa käynnillä käsitellystä sairaudesta.

Järjestelmässä käynnin diagnoosit ja käyntisyyt kirjataan samaan kenttään, ja kirjaajan roolin mukaan sitä käsitellään joko diagnoosina tai käyntisyynä. Tulosyiden kirjaamiseen ei ole määriteltyä koodistoa, vaan ne voidaan kirjata vapaana tekstinä tai vaihtoehtoisesti esimerkiksi ICPC-luokitusta (International Classification of Primary Care) käyttäen. ICPC-luokitusta on käytetty kansainvälisesti sekä tulosyiden, että diagnoosien kirjaamiseen. [14].

Diagnoosit kirjataan perusterveydenhuollossa ICD- tai ICPC-luokituksella ja erikoissairaanhoidossa ICD-luokituksella. Jos työnkulkuun otettaisiin kontaktityypin muuttavaksi tekijäksi diagnoosin kirjaaminen, työnkulku voitaisiin toteuttaa muutamalla mahdollisella tavalla: Järjestelmään voitaisiin lisätä uusi diagnoosikoodi ’Ei saapunut’. Järjestelmäteknisesti tämä onnistuisi, mutta itse keksityn koodin käyttö ei olisi kansallisten määritysten mukaista. Lisäksi tämä todennäköisesti aiheuttaisi ongelmia esimerkiksi erilaisten integraatioiden kanssa, kun integraatiosanomassa lähetettäisiin koodi, jota vastaanottavassa järjestelmässä ei ole.

Vaihtoehtoisesti ammattihenkilö voisi kirjata käyntidiagnoosin, jos käynnin syynä on ennestään tunnettu sairaus. Jos sairautta ei ole tiedossa, voitaisiin esimerkiksi käyttää ICD-10 pääluokkaan Z00-ZZB kuuluvia koodeja. Luokan sisältämät koodit ovat ”Tekijöitä, jotka vaikuttavat terveydentilaan ja yhteydenottoihin terveyspalvelujen tuottajiin” [15].

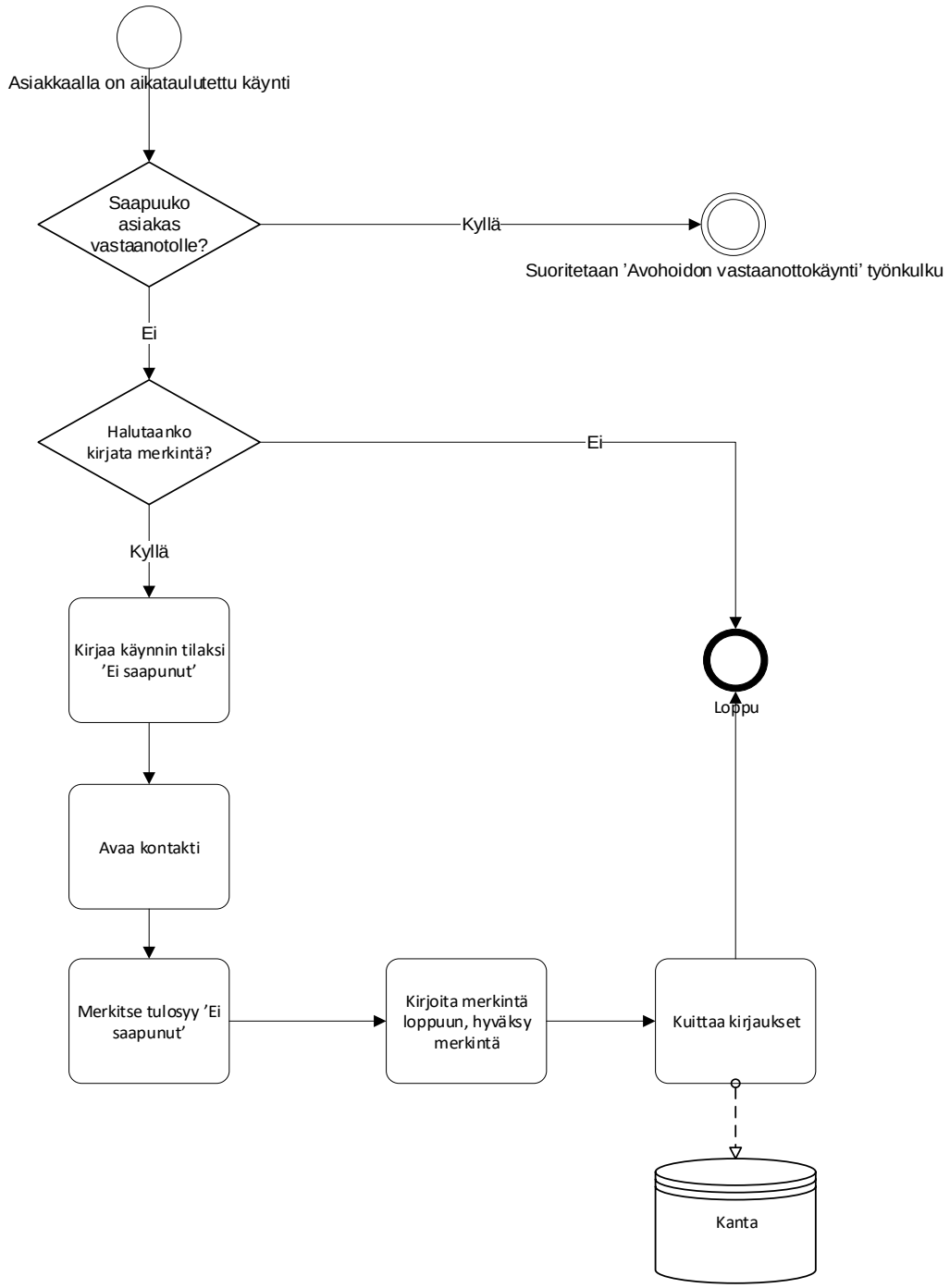
Luokasta voitaisiin käyttää esimerkiksi koodeja ’Z71.9 Määrittämätön neuvonta’, ’Z71.8 Muu neuvonta’ tai jotain tarkempaa koodia, jos käynnin syy on tiedossa. Määrittelyjen

mukaan luokan Z00-ZZB koodeja voidaan käyttää esimerkiksi, kun terve tai sairas henkilö ottaa yhteyttä terveyspalvelujen tuottajiin esimerkiksi saadakseen rajoitettua hoitoa johonkin vaivaansa [15].

Työnkulussa päädyttiin käyttämään tulosyyn kirjaamista kontaktityypin muuntavana toimintona, sillä tulosyyt eivät käytä mitään määriteltyä rakenteista koodistoa. Tämä mahdollistaa ’Ei saapunut’ -tulosyykoodin lisäämisen järjestelmään ja sen myötä sen kirjaamiselle voidaan tehdä pikapainike. Kontakti merkitään ’Ei saapunut’ -tilaan ammattihenkilön aikataulunäkymässä siihen tarkoitetulla painikkeella. Tämä toiminto on jo olemassa, joten sitä varten ei tarvinnut tehdä muutoksia.

Työnkulkua varten optimaalisin työtila sisältää merkinnän kirjoitustyökalun, sekä ”Ei saapunut” -tulosyyn kirjaamista varten pikapainikkeen. Tällainen työtila konfiguroitiin ja avattavan työtilan määrittävään sääntöön lisättiin ehtolause, joka on tosi, kun käynnin tila on ’Ei saapunut’. Ehtolauseen ollessa tosi asetetaan työnkulkua varten konfiguroitu työtila.

Työtila sisältää oletuksena kirjausten kuittaamiseen tarvittavan painikkeen. Kyseinen painike löytyy oletuksena kaikista avovastaanoton käyntien työtiloista. Kuvassa 6 on kuvattu edellä esitelty konfiguraatiomuutoksien jälkeinen saapumatta jättäneen potilaan hoitomerkinnän kirjoitustyönkulku.



Kuva 6. Konfiguraatiomuutoksilla toteutettava työnkulku.

Työnkulku alkaa aikataulutetusta käynnistä. Ei ole väliä, onko kontaktiin tehty esikirjauksia. Potilas jättää saapumatta vastaanotolle ja ammattihenkilö haluaa kirjoittaa merkin-

nän. Ammattihenkilö napsauttaa ’Ei saapunut’ -painiketta. Tämän jälkeen ammattihenkilö avaa kontaktin ja napsauttaa työtilaan avautuneesta painikkeesta tulosyyn ’Ei saapunut’. Ammattihenkilö kirjoittaa merkinnän loppuun ja kuittaa sen. Tämän jälkeen ammattihenkilö kuittaa kirjaukset, jolloin hoitoasiakirja generoituu ja lähetetään Kantaan.

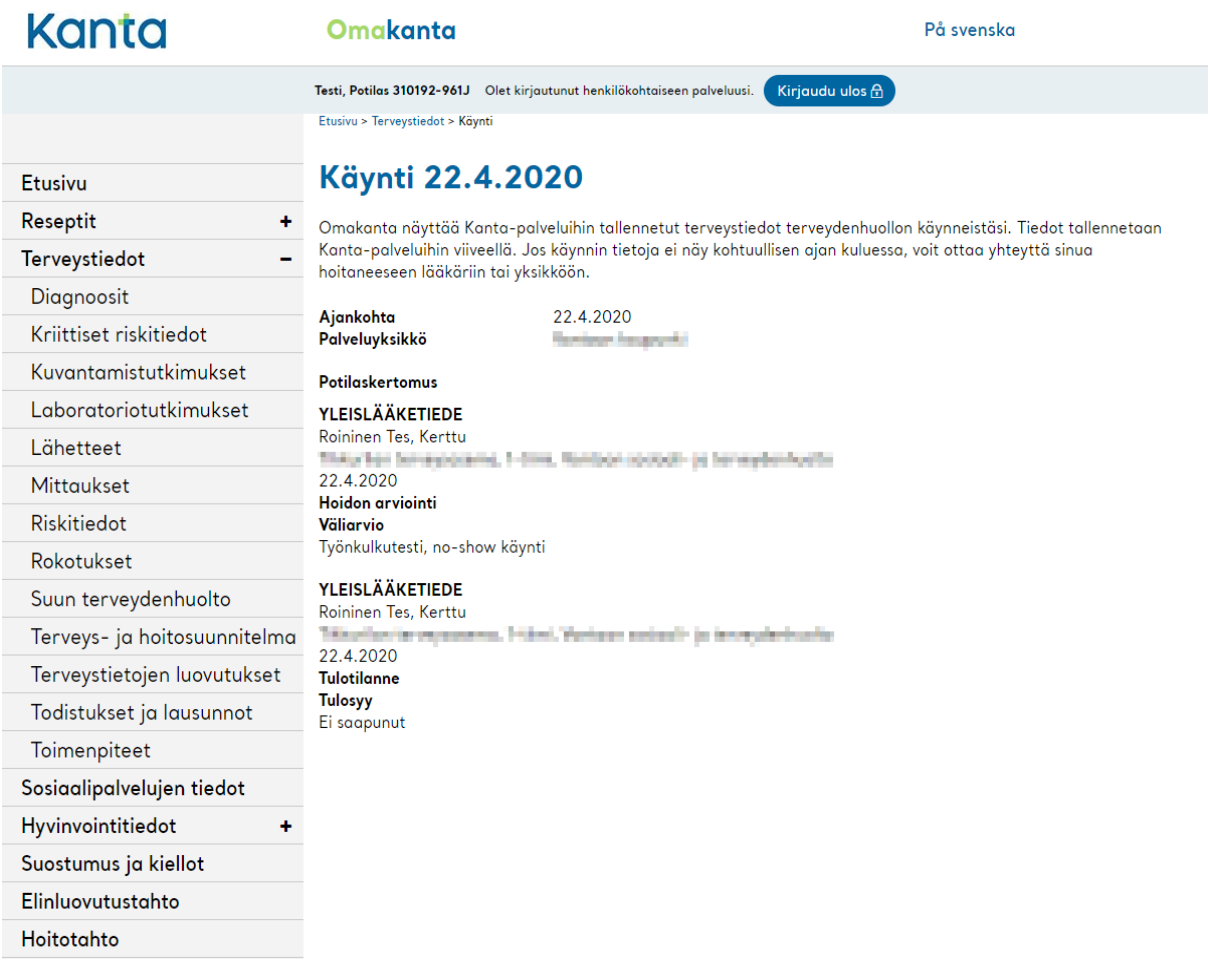
Työnkulun suorittamisen aikana järjestelmässä tapahtuu seuraavat asiat: Kun ammattihenkilö napsauttaa ’Ei saapunut’ -painiketta, kontaktin tilaksi muuttuu ’Ei saapunut. Ammattihenkilö avaa kontaktin, jolloin työtilan määrittävä sääntö osuu kohtaan, joka katsoo, onko käynnin tila ’Ei saapunut’. Ehtolause palauttaa totuusarvon tosi, jolloin järjestelmä asettaa työtilaksi työnkulkuun määritellyn työtilan. Työtilaan avautuu pikapainike ’Ei saapunut’ tulosyyn kirjaamiseen, merkinnän kirjoitustyökalu sekä käynnin kuittauspainike. Kirjausten kuittauksen jälkeen järjestelmä generoi ja lähettää hoitoasiakirjat.

6.2 Konfiguroitavan työnkulun testaus

Edellisessä kappaleessa kuvatun testiympäristöön konfiguroidun työnkulun toimivuus testattiin seuraavin testiaskelin:

1. Aikatauluta testipotilaalle käynti.
2. Tee käynnin esikirjaus, kirjoita merkintään ”Työnkulkutesti”.
3. Sulje esikirjaustila.
4. Kirjaa potilas ei-saapuneeksi.
5. Avaa potilaan kontakti.
6. Täydennä esikirjauksessa aloitettua merkintää tekstillä ”no-show käynti”.
7. Kirjaa tulosyy ’Ei saapunut’.
8. Kuittaa kirjaukset.

Testitapauksen askeleet suoritettiin järjestelmässä. Testipotilaalle aikataulutettiin käynti ja kontaktiin aloitettiin merkinnän kirjoittaminen esikirjaustyötilassa, kuten oikeassa potilaskontaktissa saatettaisiin tehdä. Tämän jälkeen käynnin kontaktista varmistettiin, että ajanvarauksen tila on ’Ei saapunut’ ja että käynnillä kirjattu merkintä näkyy Omakannassa. Kuvassa 7 näkyy testin tuloksena testipotilaan Omakantaan lähetetyt tiedot kontaktin kirjauksista.



Kuva 7. Kuva testitapauksesta Omakannassa.

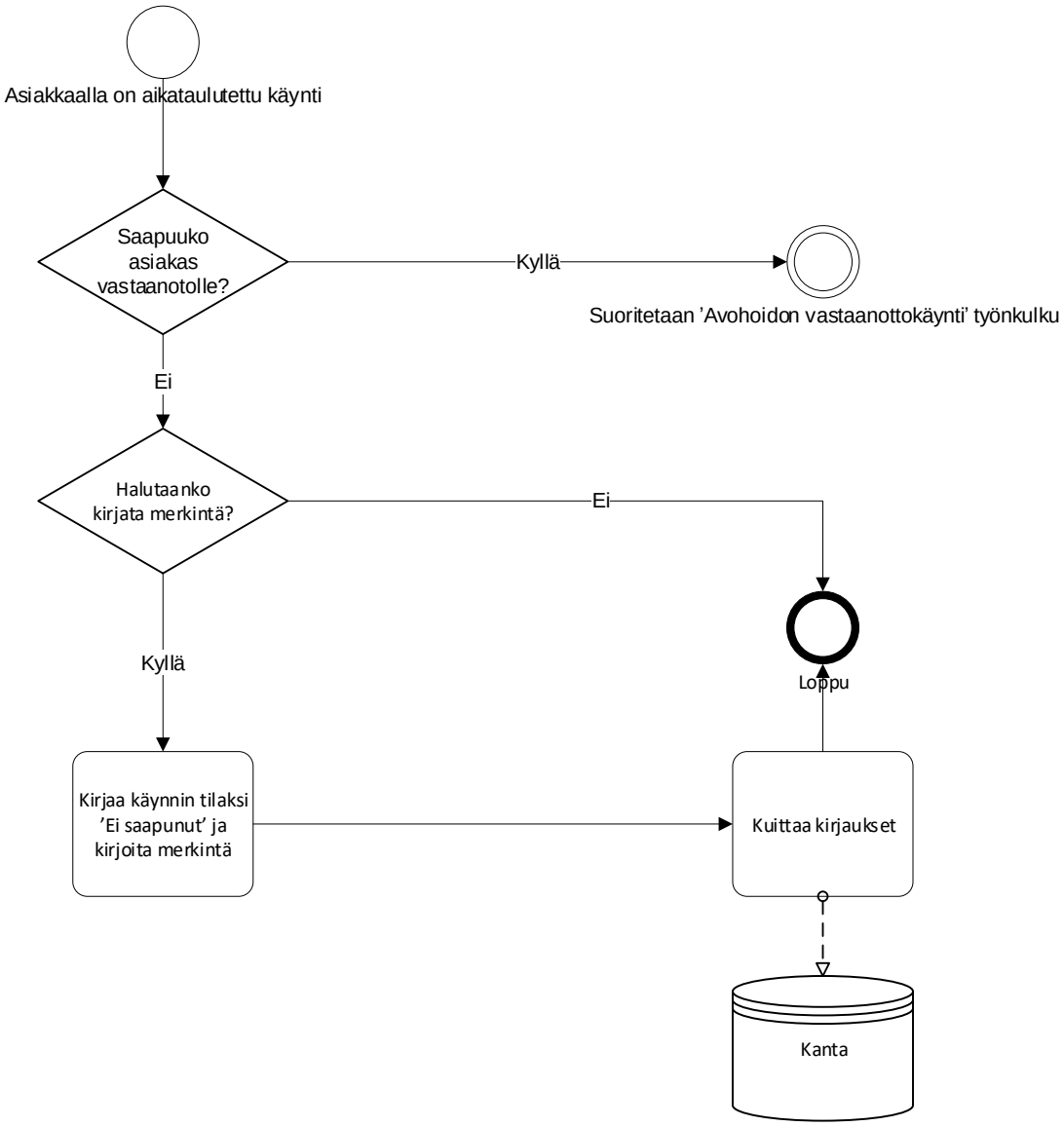
Kuvassa 7 nähdään ’Väliarvio’-otsikon alla testin aikana kirjoitettu merkintä. Tulosyy otsikon alla ’Ei saapunut’ -teksti. Testin tuloksen perusteella, voidaan siis todeta työnkulun toimivan odotetulla tavalla.

6.3 Järjestelmän kehitystä vaativa työnkulku

Edellisessä luvussa esiteltiin työnkulku, joka on mahdollista toteuttaa järjestelmään konfiguraatiomuutoksilla. Insinöörityössä toisena tavoitteena oli kehittää työnkulku, joka voitaisiin toteuttaa järjestelmän kehitystyön jälkeen. Työnkulussa pyritään hyödyntämään järjestelmän olemassa olevia toimintoja tehokkaasti, jotta kehitystyö pysyy kevyenä. Täten työnkulku ei käytännössä tarvitse uutta toiminnallisuutta - kaikki toiminnot ovat järjestelmässä olemassa, mutta ei optimaalisella tavalla konfiguroitavissa. Kuten konfiguraatiomuutoksilla toteutettavaa työnkulkua, myös sovelluskehityksen jälkeen mahdollista työnkulkua koskee seuraavat järjestelmän perustoiminnallisuuksiin nojaavat ehdot:

- Kontakti pitää merkitä ’Ei saapunut’ -tilaan.
- Kontaktityypin pitää muuntua ajanvarauksesta vastaanottokäynniksi.
- Kontakti tulee kuitata, jotta hoitoasiakirjat lähetetään potilastiedon arkistoon.

Jos työnkulku noudattaa näitä ehtoja, sitä varten ei tarvitse kehittää täysin uutta toiminnallisuutta kovinkaan paljoa, vaan voidaan pitkälti hyödyntää olemassa olevia toimintoja. Käytännössä tämä tarkoittaa sitä, että käyttäjän tekemien toimintojen pohjalta syntynyttä tapahtumankäsittelyn logiikkaa pitää muuttaa tai tehdä se konfiguroitavaksi. Työnkulku alkaa samasta vaiheesta, kuin aiemmin esitelty työnkulku: Potilaalla on aikataulutettu käynti, mutta jättää saapumatta vastaanotolle. Ammattihenkilö merkitsee aikataulunäkymässä potilaan saapumatta jääneeksi. Tässä olisi looginen vaihe kontaktityypin muuntumiselle: Kun ammattihenkilö napsauttaa ’Ei saapunut’ -painiketta, käynnin tilaksi muuttuu ’Ei saapunut’ ja kontaktityyppi muuntuu vastaanottokäynniksi, tai muuksi määritellyksi kontaktityypiksi. Lisäksi ’Ei saapunut’ -painike voisi avata ikkunan, jossa merkintä on mahdollista kirjoittaa loppuun ja kuitata kirjaukset. Työnkulku on esitelty kuvassa 8.



Kuva 8. Sovelluskehityksen jälkeen mahdollisesti toteutettava työnkulku.

Kuten konfiguroitu työnkulku, myös tämä alkaa aikataulutetusta käynnistä. Potilas ei saavu vastaanotolle, mutta ammattihenkilö haluaa kirjoittaa merkinnän. Ammattihenkilö napsauttaa 'Ei saapunut' -painiketta, jolloin kontaktityyppi muuttuu ja merkinnän kirjoitusikkuna aukeaa. Ammattihenkilö kirjoittaa merkinnän. Tämän jälkeen ammattihenkilö kuittaa kirjaukset, jolloin hoitoasiakirja generoituu ja se lähetetään Kantaan.

6.4 Yhteenveto

Insinöörityössä oli tavoitteena löytää ratkaisuja potilastietojärjestelmän hoitomerkinnän kirjoituksen työnkululliseen ongelmaan. Ongelma koski avovastaanoton käyntejä, joihin potilas jättää saapumatta. Käytössä olevassa työnkulussa saapumatta jättäneen potilaan kertomukseen luodaan uusi kontakti, johon merkintä kirjoitetaan. Käyttäjiltä tulleen palautteen perusteella on todettu, että työnkulku ei ole intuitiivinen ja työnkulkua on tarpeen kehittää. Insinöörityössä esiteltiin ongelman ratkaisuun kaksi työnkulkua: Yksi työnkulku, joka on toteutettavissa järjestelmän konfiguraatiomuutoksia hyödyntäen. Tämä työnkulku konfiguroitiin järjestelmään ja sen toimivuus testattiin. Toinen ratkaisuksi löydetty työnkulku vaatii järjestelmän kehitystyötä. Työnkulun kehittämisessä pyrittiin hyödyntämään järjestelmässä olemassa olevia toimintoja, jotta selvittäisiin kevyemmällä kehitystyöllä. Työn tulosta voidaan käyttää päätettäessä, minkälaista työnkulkua merkinnän kirjoittamiseen halutaan käyttää, sekä pohtiessa, minkälaista sovelluskehitystä järjestelmään suunnitellaan.

Lähteet

1 Valvira. Sosiaali- ja terveydenhuollon tietojärjestelmät. Verkkoaineisto. <https://www.valvira.fi/terveydenhuolto/sosiaali-ja-terveydenhuollon-tietojarjestelmat>. Päivitetty 30.3.2020. Luettu 2.4.2020.

2 Sosiaali- ja terveysministeriön asetus potilasasiakirjoista. 2009. <https://www.finlex.fi/fi/laki/alkup/2009/20090298#Pidp447550528>.

3 Terveyden ja hyvinvoinnin laitos. 2018. Potilastiedon arkiston toimintamallit. Verkkoaineisto. <https://www.kanta.fi/documents/20143/106832/Potilastiedon+arkiston+toimintamallit.pdf/39510f48-3aec-0bcb-1513-af182fc00d5d>. Luettu 27.2.2020.

4 Kanta, Potilastiedon arkisto. 2019. Verkkoaineisto. <https://www.kanta.fi/ammattilaiset/potilastiedon-arkisto>. Luettu 29.4.2020

5 Laki sosiaali- ja terveydenhuollon asiakastietojen sähköisestä käsittelystä. 2007. <https://www.finlex.fi/fi/laki/ajantasa/2007/20070159#L1P3>

6 Valvira. Potilasasiakirjat. 2018. Verkkoaineisto. <https://www.valvira.fi/terveydenhuolto/hyva-ammatinharjoittaminen/potilasasiakirjat>. Luettu 27.2.2020

7 eArkisto. 2014. Käyttötapaukset – Potilastietojärjestelmät. Liite 2 – Palvelutapahtumien esimerkkejä. Verkkoaineisto. <https://www.kanta.fi/documents/20143/107839/Potilastietoj%C3%A4rjestelmien+k%C3%A4ytt%C3%B6tapaukset+Liite2+Palvelutapahtumien+esimerkkej%C3%A4.pdf>. Luettu 20.4.2020.

8 THL, Rakenteinen kirjaaminen sosiaali- ja terveydenhuollossa. 2018. Verkkoaineisto. <https://thl.fi/fi/web/tiedonhallinta-sosiaali-ja-terveysalalla/ohjeet-ja-soveltaminen/rakenteinen-kirjaaminen-sosiaali-ja-terveydenhuollossa>. Luettu 27.10.2019.

9 Taylor, Hugh & Phillips, Les. Event Driven Architecture: How SOA Enables the Real-Time Enterprise. 2009. Addison-Wesley Professional.

10 Yeager, Dorian P. Object-Oriented Programming Languages and Even-Driven Programming. 2014. Mercury Learning & Information.

11 Stephen Ferg. Event-Driven Programming: Introduction, Tutorial, History. 2006.Verkkoaineisto. <https://sourceforge.net/projects/eventdrivenpgm/files/>. Luettu 15.1.2020.

12 Koskimies, K. & Mikkonen, T. (2005). Ohjelmistoarkkitehtuurit. Tampere: Talentum Oy.

13 Petr Kozelek. Audit Trail – Tracing Data Changes in Database. 2010. Verkkoaineisto. <https://www.codeproject.com/Articles/105768/Audit-Trail-Tracing-Data-Changes-in-Database>. Luettu 15.2.2020.

14 Jokinen, Taina & Virkkunen Heikki. Terveyden ja hyvinvoinnin laitos. Potilastiedon rakenteisen kirjaamisen opas. 2018. Verkkoaineisto. <https://thl.fi/documents/920442/2902744/Kirjaamisopas+osa+1++final+2018__.pdf/5395585e-324f-4ac5-86d6-106e27979e77>. Luettu 20.4.2020.

15 Tautiluokitus ICD-10. Terveyden- ja hyvinvoinnin laitos. 2011. Verkkoaineisto. <https://www.julkari.fi/handle/10024/80324>. Luettu 27.4.2020.